SGT1 STEbus Compatible Graphics and Text Terminal

Software Manual

C O N T E N T S

Section 1. Introduction

This manual describes the software apsects of the SGT1 board.

The Arcom SGT1 board is a low cost colour graphics board with firmware to make the generation of text and graphics displays easy. The SGT1 works in two text modes, 40- and 80-column, and three graphics modes, monochrome, 4 colour and 16 colour. The firmware adjusts the scaling of the coordinates so that the coordinate system is the same in every mode.

The on-board software provides a high level interface to the host computer.

The software has been written in a structured and extendable manner. The host can modify or replace routines and can download user-written programs via the STEbus into the SGT1 to perform any specialised operations. These programs can be integrated into the general structure of the software and can be called in the same way as the resident routines.

The board acts like any VDU display. Characters written in ASCII to the board are automatically put on the screen. Control codes like linefeed are also automatically handled. The graphics and more complicated commands are called by escape sequences.

The software includes a 63 character input buffer so that the host can send several commands to the board and then continue computing while the 6803 processes the commands. In this way the host is not held up while the board computes complex graphics commands. The read port is not buffered.

Section 2. Using the SGT1

Section 2.1 Host computer.

The SGT1 is an STEbus slave board. A processor and software will be needed to control the board.

The following combinations are just a few of the possibilities.

BASIC, assembler or C running under CP/M, CCP/M or OS-9
Stand alone C running on SCPUB or SC88
ABASIC on SCPUB

Section 2.2 Controlling the SGT1.

The SGT1 is supplied jumpered to reside at STEbus I/O locations 144 and 145 (decimal). Section 5 of the hardware manual decribes how to change this. This manual is written assuming that the address of the board is as standard. Data may be sent to or from the board through location 144. Location 145 is a register giving the status of the SGT1 board.

The SGT1 resets when SYSRST* is asserted on the STEbus.

Characters can be sent to the board directly in ASCII. Lines and other graphics functions must be sent as an escape sequence.

A test screen appears first. A read access to the SGT1 data port must be performed first to clear this. See the listing at the end of the manual for how to patch the TYPE byte to disable this screen if you wish to.

The test screen consists of some text and bars of colour. The colours are as follows, from left to right:

white cyan magenta blue yellow green red black

These bars can be used to ensure that RGB connections are correct.

It is recommended that the user first experiment using the different commands on the board using a simple ABASIC program to send bytes to the SGT1. Such a program is given here.

```
 10   PRINT " Address of SGT1";
 20   INPUT SGT
 30   PRINT "Enter data ";
 40   INPUT X
 50   IF X < 0 THEN GOTO 90
 70   OUT SGT,X
 80   GOTO 30
100   PRINT INP(SGT)
110   GOTO 30
```

If the SGT1 has standard jumpering enter 144 for the address of the SGT1. When prompted to enter data enter the decimal byte values that you wish to be sent to the SGT1. For example, entering 65 will cause a capital 'A' (ASCII code 65) to be printed on the display. Entering a negative number will make the program read data from the SGT1.

Note that the above program does not conform to the specification for writing to and reading from the SGT1. It relies on the delay involved in typing values to ensure that the SGT1 will be ready. See section 2.4 for details of using the status port to work out when to send data.

An example of a command, to draw a test pattern on the screen, is given below, with the decimal values to be entered in the above program (ie. written to I/O location 144) on the left and comments on the right:

27	Escape character
86	Indicates which escape function is required. 86 specifies "Write test pattern." Some commands require more data to follow at this point. This command does not.

This command is described in full in section 5. After writing a test pattern the command waits for a read access to the SGT1. Enter the value -1 in the program will cause the SGT1 to clear the screen and wait for the next command.

Section 2.3 The status port

Reading I/O address 145 gives the status of the SGT1 board. Only the lower four bits are significant. These bits are:

Bit 3: /BUSY
Bit 2: ERROR
Bit 1: DREADY
Bit 0: ACTIVE

BUSY is an active low signal.

The interpretation of these bits is as follows (X = don't care).

Bit: 3	2	1	0	Status
0	X	X	X	Processor is unable to process incoming data.
1	X	X	0	Reserved
1	X	X	1	Host may write data.
1	0	1	1	Data is available to be read from SGT1.
1	1	1	1	Data on output port is an error code.

See section 7 on error codes.

Section 2.4 Accessing the SGT1

Algorithms and example BASIC subroutines are given here to show how to access the SGT1.

The algorithm to write data to the SGT1 is as follows.

```
Wait for NOT BUSY AND ACTIVE
THEN wait 20us
THEN write data
No further access must occur for 50us
```

A BASIC subroutine implementation of this, where the SGT1 is at its standard I/O address and the value to be written is passed in X, is given here.

```
2000 IF ( INP(145) AND 9 ) <> 9 THEN GOTO 2000
2010 FOR I = 1 TO 5
2020 NEXT I : REM Delay. BASIC is slow enough without this loop
2030 OUT 144, X
2040 RETURN
```

An algorithm to read data from the SGT1 is as follows.

```
Wait for DREADY AND NOT BUSY AND ACTIVE
THEN wait 20us
IF data is an error code then handle error and try again
THEN read data
No further access must occur for 50us
```

A BASIC subroutine implementation of this, where the value read is returned in X, is given here.

```
1000 IF ( INP(145) AND 11 ) <> 11 THEN GOTO 1000
1010 FOR I = 1 TO 5 : NEXT I : REM Delay
1020 IF ( INP(145) AND 4 ) THEN GOSUB ERROR_HANDLER : GOTO 1000
1030 X = INP(144)
```

1040 RETURN

Note that once an escape sequence has been started, the host must send the whole escape sequence before any other commands, such as printing a character, can be sent. All commands are described in Section 5.

Multiple commands

There are only a few times at which the SGT1 will accept commands. These are described below.

There are two ways to get the SGT1 to execute multiple commands without needing the constant attention of the host processor - by putting a series of commands into the SGT1's input buffer and by instructing it to execute a command sequence held in its memory.

To submit a command sequence to the SGT1 use the "write to block of memory" command to set up the command sequence in the 6803 memory. Then use the "submit" command to instruct the SGT1 to execute the sequence.

Note that if a command in the sequence sends data back to the host then the host must read this data before the SGT1 will process the following commands. The command sequence will be inserted in the input stream immediately after the "submit" command. If the last command of the sequence is not a complete command then the SGT1 will take data from the input buffer to complete the command. Commands are taken directly from the input buffer as soon as the submit sequence has ended. The command sequence can be up to 255 bytes long and it is possible to chain another submit command on the end, though it is not possible to nest sequences.

Apart from releasing the host processor from sending commands individually, one other use of this command is that a sequence of graphics commands to draw, for example, a square using relative graphic coordinates could be loaded into the SGT1. The host need only position the graphics cursor and tell the SGT1 to execute this command sequence for a square to be drawn. The square routine can be used several times to draw the same feature in different places on the screen.

Normally commands to the SGT1 are not buffered. If the SGT1 has a command to process it asserts BUSY and will not accept bus accesses. There are two ways to put data into the SGT1's input buffer - writing data when the SGT1 is waiting for a read access or using the "batch" command.

The "batch" command allows the host to write a series of bytes into the SGT1's input buffer. The SGT1 will then execute these commands without requiring the attention of the host. This is an improvement on the normal method of having to wait for the previous command to finish before the SGT1 will accept the next. Note that if one of the commands in the batch sends data back to the host then the SGT1 will wait until the host has read the data before continuing.

Some commands require the SGT1 to send data back to the host. When this happens the SGT1 goes into a wait-for-interrupt state and waits for a bus access. The access should be a read. If the host makes a write access the

SGT1 will accept the data, which it will put into its input buffer, and will wait for a read access again. Thus the SGT1 input buffer can be filled with a stream of commands before a read access is performed in much the same way as the batch command works.

The SGT1 has a buffer of 63 characters. The host must take care not to overfill this buffer. The only signal available to indicate how full the buffer is, is that the buffer is empty when the SGT1 is not executing a "batch" command and /BUSY and ACTIVE are asserted in the status port.

Useful commands are:

ESC 85	Repeat next command
ESC 88	Wait for "read" access
ESC 94	Submit command sequence
ESC 126	Batch command

Section 3. Graphics Features

Section 3.1 The Screen

The way that video refresh RAM is mapped onto the screen is described in section 6.9.

The default screen size is 160 by 32 rasters, though most screens can cope with a larger display. The size can be changed from the host.

Different monitors may require slightly different values in the 6845 registers. Details on the registers are given in the hardware manual.

Features such as setting the mode and the screen size should be done at reset to avoid confusing the software.

Features drawn on the screen will be in one of three colours. All text will be printed in the text colour. Graphics features will be plotted in the graphics plotting colour unless the plotting mode is changed (see section 3.5). The background colour is used for:

Clearing the screen
The background to text
AND plotting graphics features (see section 3.6)
Erasing text lines and characters

Useful commands are

ESC 96	Set background colour
ESC 101	Set colour mode
ESC 104	Set 6845 register
ESC 106	Set screen size
ESC 107	Set display base address
ESC 108	Read display base address

Section 3.2 Graphics Coordinate System

Graphics functions are plotted using the graphics coordinate system. The origin is the bottom left hand corner of the screen. The screen is considered to be 1280 pixels wide and 1024 pixels high, unless the screen size is changed. This arrangement applies to all colour modes. Because in different colour modes the resolution of the video signal is less than this, the pixel coordinates are scaled down by the software. To allow for this each physical pixel on the screen corresponds to more than one logical pixel. Thus plotting a point at coordinates (X,Y) plots a point at the same place in all modes.

A physical pixel on the screen is four logical pixels high, and, in mode N, is N pixels wide. Thus in mode 2 logical points (0,0), (0,1), (0,2), (0,3), (1,0), (1,1), (1,2), (1,3) are all mapped onto the same physical pixel.

A non-displaying graphics cursor is provided. This can be moved about the screen and most graphics commands use the position of the graphics cursor

as one point to define the particular graphics feature.

Most graphics functions need two points to be defined. For example to draw a line the start and end points must be defined. The start of the line is defined as the current graphics cursor position. The other point can be defined in two ways, absolute or relative coordinates.

Absolute coordinates is where the second point is defined using an (X,Y) coordinate pair. The origin is the bottom left hand corner of the screen.

Relative coordinates is where the second point is defined as an offset from the current graphics cursor position. Thus the series of HEX bytes

ESC 83 00 00 80 01 00

draws a line from the current graphics cursor to the point (80h,100h), whereas

ESC 83 01 00 80 01 00

always draws a line 100h pixels upwards and and 80h pixels to the right to the right of the current graphics position.

Section 3.3 Text Display

The SGT1 can be used as a text display, with all the standard control commands available. The control codes are standard (eg. 13 is carriage return) and there are also some escape sequences for functions such as insert or delete a line. These are described in section 4. The escape sequences have been chosen to be the compatible as for the VDU3 sold by Arcom, where ever possible.

All the functions required to run programs such as Wordstar are implemented. Arcom can supply a driver for CP/M so that the SGT1 can be used as the main console display.

Section 3.4 Initialisation and Defaults

After reset, applications may need to set up some of the following features of the SGT1. The feature, the default and which command changes it are given.

Feature	Command	Default
Mode	ESC 101	Mode 4
Screen size	ESC 106	160 * 32
6845 registers	ESC 104	Typical values given in Hardware manual
Text Cursor	ESC 46	Off

Graphics drawing colour	ESC 109	White
Text print colour	ESC 95	White
Text background colour	ESC 96	Black
Plot mode	ESC 124	OR
Dash mode	Unimplemented	No dashing

After changing the mode the print, plot and background colours will need to be set. After changing the screen size any picture on the screen at that time will become distorted.

3.5 Scrolling

The SGT1 can scroll the display up, down, left and right. All scrolling is performed on whole rasters. Vertical scrolling moves by the height of one character line at a time. Horizontal scrolling moves by a quarter of the width of a character drawn in mode 4.

The video refresh RAM available is larger than the amount needed for the screen. By changing the block of memory used to refresh the screen the effect of scrolling can be achieved. Thus moving the block up through memory by 8 locations gives left scrolling. The only problem with this scheme is that repeated scrolling in one direction will cause the block to reach the limit of the video refresh RAM. When this happens the screen contents are copied to the other end of video RAM to allow scrolling to continue. The copying takes a short time and makes the display jump slightly.

Horizontal scrolling is achieved by using explicit commands provided. Vertical scrolling is implicitly achieved by manipulating text. Sending a linefeed when the text cursor is on the bottom text line will cause the display to scroll upwards. Inserting a text line at the top of the screen will cause a downwards scroll.

Two other scrolls are provided. These scroll a contiguous block of memory up or down eight places. On the screen this gives the effect of a block scroll. These commands were provided to scroll a segment of a single line. They will not scroll a block in the graphics sense.

With all scrolls the new part of the screen is cleared to the screen background colour.

Relevant commands are:

ESC 42	Scroll memory block left
ESC 43	Scroll memory block right
ESC 67	Scroll left
ESC 68	Scroll right
ESC 69	Insert line
10	Line feed

3.6 Plotting

Three modes of plotting are provided: XOR, OR and AND.

XOR is a non-destructive plot. All plotted bits are inverted so that replotting the same feature in the same colour will restore the original picture.

OR plotting will plot in the plotting colour.

AND plotting will plot in the screen background colour.

Useful commands:

ESC 83	Draw graphics feature
ESC 96	Set background colour
ESC 109	Set plotting colour
ESC 124	Set plotting mode

3.7 Status Line

Many 24 line VDUs provide a 25th line at the bottom of the screen which can be used as a status line. This cannot be written to in the normal way and the VDU ignores it when scrolling. This line can used to provide a status report, either on the terminal itself or the computer attached to it.

The SGT1 firmware provides such a status line. Unlike most VDUs if the status line is disabled the display uses the bottom line as part of the graphics or text display. When the status line is enabled the bottom raster of the display is cleared and cannot be used for normal text. On reset the status line is disabled.

To write to the status line an escape sequence followed by a string of text must be sent. The text can be terminated with a carriage return, line feed or when the status line is full. The length of the status line is one less than the number of characters that can be displayed across the screen. The status line is written in inverse video. The status line must be turned on before writing to it.

The operation of the status line is slightly non-standard because turning the status line off and then on again will lose the text previously on the status line.

The commands provided to use the status line are:

ESC	102	Write to status line
ESC	103	Turn status line on
ESC	125	Turn status line off

Section 3. Command Summary

This section is designed to be an index into Section 4 where these commands are described in full.

Section 3.1 General Commands

ESC 38	Set palette colour
ESC 39	Set whole palette
ESC 64	Write a byte to memory or device
ESC 65	Load data into memory
ESC 66	Read data from 6803 memory
ESC 67	Scroll left one raster
ESC 68	Scroll right one raster
ESC 86	Display test pattern
ESC 88	Wait for read operation
ESC 92	Read a byte from memory or device
ESC 94	Submit command sequence
ESC 96	Set background colour
ESC 99	Clear Screen
ESC 101	Set colour mode (mono, 4- or 16-colour) *
ESC 104	Set a 6845 register
ESC 106	Set up screen size (height and width)
ESC 107	Set display base address
ESC 108	Read display base address
ESC 126	Receive command sequence
ESC 127	Warm start

Section 3.1 Control Codes

CTRL H	(8)	Backspace
CTRL I	(9)	Tab
CTRL J	(10)	Linefeed
CTRL K	(11)	Cursor up
CTRL L	(12)	Cursor right
CTRL M	(13)	Carriage return
CTRL P	(16)	Cursor down
CTRL ^	(30)	Cursor home
CTRL _	(31)	Newline

Section 3.2 Text Escape Sequences

ESC 40	Half intensity off
ESC 41	Half intensity on
ESC 42	Scroll block left
ESC 43	Scroll block right
ESC 46	Set cursor style
ESC 61	Move cursor
ESC 63	Read text cursor position
ESC 69	Insert line
ESC 71	Set printing style

ESC 81	Insert character
ESC 82	Delete line
ESC 84	Erase to end of line
ESC 87	Delete character
ESC 89	Erase to end of page
ESC 91	Set cursor row address
ESC 93	Set cursor column address
ESC 95	Set printing colour
ESC 102	Write to status line
ESC 103	Status line on
ESC 125	Status line off

Section 3.4 Graphics Escape Sequences

ESC 83	Graphics functions: draw line move cursor plot point read pixel draw arc block fill
ESC 90	Step graphics cursor
ESC 97	Read graphics cursor position
ESC 109	Set graphics drawing colour
ESC 124	Set graphics plotting mode (OR, AND, XOR)

Section 4. Commands

This section describes all the resident commands. A command summary is given in Section 3. The escape character is decimal 27. Control codes are those characters with values less than 32. Note that if a control character is sent in the middle of an escape sequence it will be interpreted as a part of the escape sequence.

A list of control codes and their effects is given in Section 3.

Below the title of each command is a line starting ESC ... . This line gives the values that must be sent to the board to achieve the effect. The quantities in angled brackets are arguments to the command. They must be replaced with values of the appropriate type. The arguments are as follows.

<address>	Double byte value, indicating a 6803 memory location. The more significant byte must be sent first.
<byte>	Single value.
<X-coord> <Y-coord>	Double byte value giving absolute coordinates for use in the graphics coordinate system.
<X-disp> <Y-disp>	Double byte value giving a displacement from the current graphics cursor position.

For example if a command requires an address, sending the hex numbers B0 and 80 will specify address 0B080h. Two-byte values are always sent high byte first.

All other argument names are specific to one command and are single byte values unless otherwise stated.

For example escape 104 sets up a 6845 register. The command sequence given is:

ESC 104 <reg> <value>

The values within angled brackets are parameters and must be replaced by the quantity they represent.

To use this command to set register 2 to 170 write the following 4 bytes to STEbus I/O location 144.

27, 104, 2, 170.

Set Palette colour

ESC 38 <log col> <phys col>

This command is only applicable when using the SGTX extension board to the SGT1. The palette colour <log col> is assigned the physical colour <phys col>.

Set Whole Palette

ESC 39 <data>

This command is only applicable when using the SGTX extension board to the SGT1. Re-assigns the whole palette. The data consists of 16 bytes. The first byte is written into palette colour 0 etc. up to palette colour 15.

Half Intensity Off

ESC 40

Sets text printing to normal intensity. On the current version of the software this has the effect of setting the print colour to white (palette colour 3 in mode 2).

Half Intensity On

ESC 41

Sets text printing to half intensity. On the current version of the software this has the effect of setting the print colour to red (palette colour 1 in mode 2).

Line scroll Left

ESC 42 <offset> <size>

Rotates any block of memory eight bytes down memory. When performed on video RAM this appears to scroll a block of memory leftwards. The block is specified by the offset (from the bottom of display memory) and size parameters. No memory outside this block is altered. The top eight bytes of the block are cleared to the screen background colour. The command is designed to scroll small segments of a single text line.

Size is a 16-bit quantity.

See ESC 67 and ESC 68 for scrolling the whole screen.

Line scroll right

ESC 43 <offset> <size>

Rotates any block of memory eight bytes up memory. When performed on video RAM this appears to scroll a block of memory rightwards. The block is specified by the offset (from the bottom of display memory) and size parameters. No memory outside this block is altered. The bottom eight bytes of the block are cleared to the screen background colour. The command is designed to scroll small segments of a single text line.

See ESC 67 and ESC 68 for scrolling the whole screen.

Set Cursor Style

ESC 46 <style>

Determines whether the cursor is on or off.

'0' Cursor off
'2' Cursor on, not flashing

Note that the bytes to send are the printing characters '0' and '2', ASCII values 48 and 50 respectively.

Move Text Cursor

ESC 61 <row> <column>

Move text cursor to specified row and column. The row and column arguments are single bytes. An offset of 32 must be added to the coordinates. Thus the top left hand corner is accessed as (32,32).

Read Text Cursor Position

ESC 63

Allows the host to find out where the cursor is. After receiving this command the SGT1 sends two bytes back to the host: the first byte is the row address and the second the column address. Both have an offset so that the home position is returned as (32,32).

Write a byte to 6803 memory.

ESC 64 <address> <byte>

Write a byte to a 6803 memory address. The 6803 can also access on-board I/O devices at specified memory locations (see section 6) and these may also be written to.

Note that the software keeps copies of the values it writes to some devices so not updating these copies may confuse the software (see listing for SGT1 workspace RAM).

See also the read a byte command (ESC 92).

Load data into memory

ESC 65 <address> <count> <data>

Load a stream of data into 6803 memory. The count gives the number of bytes of data to follow. Count is a two-byte value. The data is loaded from the given address and consecutive addresses above this.

Dump data from memory.

ESC 66 <address> <count>

Requests the 6803 to send <count> bytes of data to the host computer, read from the given address upwards. Count is a two-byte value.

Scroll leftwards

ESC 67

The whole screen is screen is scrolled leftwards by one raster, ie 8 logical pixels. The new part of the screen is cleared to the background colour.

Scroll Rightwards

ESC 68

The whole screen is screen is scrolled righhtwards by one raster, ie 8 logical pixels. The new part of the screen is cleared to the background colour.

Insert Line.

ESC 69

Insert a text line at cursor position. The line at the cursor and lower lines are moved down, the current line blanked and the cursor returned to the beginning of the line.

Set Printing Style

ESC 71 <style>

Determines what operations are performed on the text before printing. If bit 1 of the "style" byte is set the text is in inverse mode, ie. background and foreground swapped.

Insert Character.

ESC 81

Move all characters from cursor position to end of line right one place, losing last character, and prints a space at cursor position.

The cursor position does not change.

Delete Line.

ESC 82

Delete line with cursor on. Move all lower lines up one and blank bottom line.

The cursor returns to the beginning of the line.

Graphics command

ESC 83 <function> <X-coord> <Y-coord>
ESC 83 <function> <X-disp> <Y-disp>

This escape sequence is used to call most of the basic graphic functions available. Even function numbers imply that the X and Y parameters are absolute coordinates and odd function numbers imply that the parameters are displacements from the current graphics cursor position.

ESC 83 0 <X-coord> <Y-coord>
ESC 83 1 <X-disp> <Y-disp>

Plot a line in the current plotting colour using the graphics coordinate system from the current graphics cursor. The line is dashed according to the dash mode (default is a solid line). Lines are clipped so if any part is outside the limits of the screen that part is not drawn.

The graphics cursor is updated to the other end of the line drawn. Note that if the line is clipped the graphics cursor may end up pointing to non-displayed coordinates.

ESC 83 10 <X-coord> <Y-coord>
ESC 83 11 <X-disp> <Y-disp>

This is a quick line drawing algorithm.

Plot a line in the current plotting colour using the graphics coordinate system from the current graphics cursor. The line is not dashed, the plot is XOR and no check is made to ensure that the line is on the screen.

The graphics cursor is updated to the other end of the line drawn.

ESC 83 2 <X-coord> <Y-coord>
ESC 83 3 <X-disp> <Y-disp>

The graphics cursor is moved to the given position. Nothing is plotted.

ESC 83 4 <X-coord> <Y-coord>
ESC 83 5 <X-disp> <Y-disp>

Plot a point in the current plotting colour using the graphics coordinate system from the current graphics cursor. The point will not be plotted if it lies outside the limits of the screen.

The graphics cursor position is not changed.

ESC 83 6 <X-coord> <Y-coord>
ESC 83 7 <X-disp> <Y-disp>

Read the colour of a pixel on the screen. Returns the palette colour of the pixel. The SGT1 will not execute any more commands until the host has read the data port.

ESC 83 8 <X-coord> <Y-coord> <angle>
ESC 83 9 <X-disp> <Y-disp> <angle>

Draw an arc. The centre of the arc is the current graphics cursor position. The arc is drawn clockwise from the point specified by the coordinates or displacement through the required angle.

The angle is a two byte value. The first byte gives the angle of the arc in radians. The second byte gives the fraction of a radian in 1/256 parts of a radian. Thus an angle of 0646H would draw a full circle because

(6 + (046H / 256) = 2 * PI)

For example, to draw a complete circle radius 256 pixels about the point (512,512) move the graphics cursor to the point (512,512) then send the bytes

27 83 9 1 0 0 0 6 72

Note that the radius is set by defining a displacement to a point 256 pixels away.

The current algorithm draws exactly <angle> pixels so that small radius arcs will appear thicker than large radius arcs. XOR plotting does not give very good results for arcs.

ESC 83 12 <X-coord> <Y-coord>
ESC 83 13 <X-disp> <Y-disp>

Fill a rectangular block. A rectangular block with one corner specified as the current cursor position and the opposite corner defined by the coordinates in the command is filled with the current graphics plotting colour. The plot can be OR, XOR or AND. The block is trimmed if it is partly off the screen. The block does not need to be aligned on a raster boundary.

Delete To Right Of Cursor

ESC 84

All characters to the right, including the cursor position, are erased. The cursor does not move.

Print Test Pattern

ESC 86

A test pattern is displayed on the screen. This is to help in setting up the display and ensure that the bus interface is working.

The version and issue numbers and other messages are displayed on the screen. The SGT1 then waits for a read access from the host.

Delete Character.

ESC 87

Delete character at cursor position, move all characters to right of cursor left one place. Print space at end of line.

Wait For Read Access

ESC 88

On executing this command the SGT1 will wait until the host performs a read access before continuing. This can be used to synchronise operations on the two processors.

Erase To End Of Page

ESC 89

All characters from the cursor position to the bottom of the screen, including the cursor position are cleared to spaces. The cursor does not move and the status line is not affected.

Step Graphics Cursor Position

ESC 90 <direction>

This command steps the graphics cursor in any of eight directions. The size of the step is the width or height of one physical pixel in mode 4. A pixel is plotted at the new location in XOR mode.

The SGT1 does not check that the graphics cursor is on screen before plotting, so that plotting an off screen location may crash the board.

The direction parameter is as follows:

0	No move (but plot pixel)
1	Left and up
2	Left
3	Left and down
4	Down
5	Right and down
6	Right
7	Right and up
8	Up

Set Cursor Row Address.

ESC 91 <row>

Move cursor to specified row. Cursor column address does not change. An offset of 32 must be added to the row, ie. sending row = 32 moves to the top row of the screen.

Read a byte from 6803 memory

ESC 92 <address>

Read a byte from a 6803 memory address. This can be any device that the 6803 can access.

See also the write a byte command (ESC 64).

Set Cursor Column Address.

ESC 93 <column>

Move cursor to specifed column. Cursor row address does not change. An offset of 32 must be added to the column, ie. sending column = 32 moves the cursor to the left hand column.

Submit A Command Sequence

ESC 94 <start addr> <count>

Having loaded a sequence of bytes into 6803 memory that correspond to the bytes for a sequence of commands this command will instruct the SGT1 to execute those commands. The bytes of the command sequence are inserted in the input stream immediately after the submit command. If the last command of the command sequence is not a complete command then the SGT1 will complete the command using the next bytes from the host.

Submit sequences cannot be nested.

Set Printing Colour 0 - 3 0 = BLACK

ESC 95 <colour> 0 - 15

Sets the colour in which characters are printed. The colour parameter sholud be in the range 0 to the value of the graphics mode.

Set Background Colour

ESC 96 <colour>

Sets the background colour for the board. When characters are printed the background of the character location is set to this colour. Clearing the screen makes the screen this colour. AND plotting graphics will plot in this colour.

The colour parameter should be in the range 0 to mode, the current graphics mode.

Read graphics cursor position

ESC 97

The 6803 sends four bytes of data to the host computer; X coordinate then the Y coordinate of the graphics cursor, high byte first in both cases. See Section 2 for how to read this data.

Erase Screen

ESC 99

Clear whole screen to palette colour 0.

Set Graphics mode

ESC 101 <mode>

Set up the graphics mode for the SGT1 board. The allowed values for mode are 1,2 and 4. Note that the numbers of columns of text is also changed by this command.

Other on-board features must be reset after changing mode. For example the coordinates of the text screen do not change but the corresponding on-screen location will change.

Write To Status Line

ESC 102 <text>

If the status line is enabled the text is written to the status line. If it is disabled the text is read but not displayed. The text is terminated by a carriage return or a line feed or when the status line is full. The length of the status line is one less than the width of the screen.

The text is written in inverse colours. The background and printing colours are reversed.

Enable Status Line

ESC 103

Enables status line. The bottom line of the screen is cleared to the printing colour.

Set a 6845 register

ESC 104 <reg> <val>

Write to a 6845 register. The typical values are given in the hardware manual.

To change the number of displayable characters use the set up screen size command (ESC 106). To change the base address of video RAM use the set base address command (ESC 107).

Set up screen size

ESC 106 <SCR.WID> <SCR.LEN>

The two parameters give the displayable width and height of the monitor used. Both parameters are single bytes.

SCR.WID is two times the number of text characters displayable in mode 2 or 1, typically 160.

SCR.LEN is the number of text lines displayable, typically 35.

This command writes to two 6845 registers and sets up a number of variables concerned with printing characters and plotting points within the software.

Set Display Base address

ESC 107 <address>

This sets up the 6845 base register and software variables. This value changes after a text scroll. Typical values are $2000 to $3000.

See also read base address command (ESC 108).

Read Display Base address

ESC 108

Sends the display base address to the SGT1 read port, high byte first. The software will wait until both bytes of data have been read before executing the next command.

See also set base address command (ESC 107).

Set graphics drawing colour.

ESC 109 <value>

Sets up the palette colour in which lines and points are plotted.

Set Graphics plotting mode

ESC 124 <mode>

Three modes of plotting are provided.

XOR plotting will XOR the image on the screen with the plotting colour at all points to be plotted. This means that re-plotting the same line will restore the original image.

OR plotting will draw in the plotting colour.

AND plotting will plot in the character background colour.

The mode byte puts plotting into the following mode.

mode	Plotting mode
0	No plotting
1	XOR
2	OR
3	AND

The plotting colour can be changed with ESC 109 and the character background colour can be changed with ESC 96.

Status Line Off

ESC 125

Disables and clears status line. The bottom line is used as a normal text line.

Receive Command Sequence

ESC 126 <count> <bytes>

The SGT1 will wait until it has read <count> bytes from the host processor before executing any more commands. The bytes read are put into the input buffer and will be executed as commands. This command ignores bytes already in the input buffer.

<count> is a single byte value. Its value can not be more than 63. The user must ensure that the input buffer does not overflow.

Cold Start

ESC 127

Perform software reset.

Section 5. Memory Map and On Board Devices

Section 5.1 Memory Map.

0000h - 007Fh	6803 Internal devices
0080h - 00FFh	Zero page on chip RAM
0400h	6845 register select
0401h	6845 data
0800h	Read/write data to STEbus data port
0C00h - 0C0Fh	Palette. ADDSEL must be low to access this
1000h - 17FFh	RAM for user
1800h - 1FFFh	Workspace RAM, including escape function table, control function table, character font.
2000h - DFFFh	Video RAM
E000h - FFFFh	EPROM or RAM.

Section 5.2 6803 Internal devices.

All the 6803 internal devices are describes in the 6803 technical manual. All are available to the user except the two I/O ports. These are described here.

Port 1, bit 0:	Reserved.
Port 1, bit 1:	Input. Type. High if last STEbus access was a read. Low if last access was a write.
Port 1, bit 2:	Output. BUSY signal to status port. Low if processor is in a WAI state, else high.
Port 1, bit 3:	Output. Selects which half of EPROM is switched in when using a 27128. Normally high, this signal is fed directly into A13 on 27128.
Port 1, bit 4:	Output. ADDSEL, set low before writing to palette. High for normal operation.
Port 1, bit 5:	Output. MODSEL2, selects palette
Port 1, bit 6:	Output. MODSEL1, 0 = modes 1 & 2, 1 = mode 4
Port 1, bit 7:	Output. /ERROR signal to status port.

Port 2, bit 0:	Output. /ACTIVE signal to status port. When high, all bus accesses will hang. When low, bus accesses will send an NMI to the 6803.
Port 2, bit 1:	Reserved
Port 2, bit 2:	Output. /DREADY signal to status port
Port 2, bit 3:	RDATA
Port 2, bit 4:	TDATA

Section 5.3 Zero page RAM.

128 bytes of zero page RAM are available and have been assigned as declared in the listing in Appendix 1. User's routines should use the workspace WORK to WORK + 31, though some subroutines will also use some of these locations. Some escape commands change variables. Others may be changed using the Put-byte command (ESC 64).

A line that has an asterisk at the end in the listing of the firmware supplied with this manual indicates a variable that may be sensibly altered by application programs.

Section 5.4 6845 CRT Controller

To access data to the 6845, write the register number to location 0400h then read or write to location 0401h. The hardware manual gives the function of each register. Note that registers 1, 6, 12, 13 should be set using the specific commands. see commands ESC 104, 105, 106, 107, 108.

Section 5.5 STEbus read/write port.

Writing to 0800h will put data into the STEbus read latch. This may then be read by the host processor. Reading 0800h will return the data being written to the board by the host. This value is not latched in hardware so data must be read before the host receives a DATACK*.

The status register is set up using port 1 and port 2 on the 6803.

Section 5.6 Palette.

Requires the SGTX board (not yet available).

Section 5.7 User RAM.

RAM from 1000h to 17FFh is reserved for users' programs and data.

Section 5.8 Workspace.

RAM between 1800h and 1FFFh is defined as in the listing in Appendix 1.

ESC.TBL and CHAR.TBL are tables of two-byte addresses of subroutines.

Locations ESC.TBL and ESC.TBL+1 hold the address of the routine to handle the sequence 'ESC 0 ...'. All escape routines are accessed by indexing into this table. Similarly locations CTRL.TBL and CTRL.TBL+1 hold the address of the routine to handle control code 0.

The character font is stored in RAM at CHAR.TBL. This may also be changed by the user. Each character is stored as 8 bytes. The low address byte of each character font represents the top line of the character and the high address byte the bottom line. The most significant bit of each byte is written to the left hand end of the character. The font for ASCII character 0 is stored at the low address.

Application programs may change the character font and escape and control vector tables.

Section 5.9 Video Refresh RAM.

Video refresh RAM is from 2000h to DFFFh. Not all of this is used at once. The part that is changes during a text scroll.

The video refresh RAM is organised into rasters of eight bytes. The lowest video RAM address is mapped onto the top left of the screen. The next seven bytes are directly below this. These eight bytes are the first raster. The next eight bytes make the second raster which goes to the right of the first one.

1 raster = 8 × 8 pixels

Section 5.10 EPROM.

Memory from E000h to FFFFh can be one of two banks of a 27128 EPROM or RAM. Write operations always access the RAM but reads depend on MAP (Port 1, bit 0) and PROMBANK (Port 1, bit 3).

MAP	PROMBANK	Memory accessed on a read
0	X	RAM
1	0	Low half of 27128
1	1	Top half of 27128
1	X	2764

Section 5.11 EPROM Structure.

Version number, issue number and routine jump tables are declared as in the listing in Appendix 1.

Section 7. Running programs on the SGT1

The following information will be useful to users who wish to run programs on the SGT1 itself.

Some specialised applications may require this for reasons of speed or for flexibility. Programs are incorporated into the structure of the SGT1 and can be called using an escape sequence. Parameters can be passed.

Appendix A is a listing of the beginning of the firmware on the SGT1. The firmware variables and the routines marked with an asterisk on the right hand side are implemented, debugged and available to be called and modified by the user.

An area of RAM is reserved for the user. This is described in section 6.

The escape sequences ESC 0... to ESC 31... are reserved for the user. To call a program the address to call must be loaded into the escape table at an entry corresponding to one of these sequences.

The actions involved in loading and running a program are as follows.

Program stored as data in the host machine
Download program into user RAM on SGT1 using write data command-ESC 65
Patch escape table so that an escape sequence will call the program
Send escape sequence to call routine.

On version 1, issue 3 of the software interrupts are disabled.

Appendix. Data Structures

The bottom of the EPROM supplied is of the following form. Note that entries in the jump table will always be in the same place so that users' downloaded routines may vector through this table and be compatible with all versions of SGT1 software.

The routines, variables and data structures marked with an asterisk on the right hand side are implemented, tested, documented and usable by application programs.

```
                ; (C) Copyright Arcom Control Systems Ltd. 1985

                ; P. Bhagat 19-Jun-85


0001            VERS    EQU     1
0003            ISSUE   EQU     3

                ; The software determines this byte, not vice versa
                ; TYPE
                ; One byte giving the characteristics of the software
                ;       bits 7,6,5: Configured emulation
                ;            0 0 0:     Ampex210
                ;
                ;       bit 4: 1 = Write test display at power-up
                ;              0 = Disable test dispolay

0010            TYPE    EQU     $10

001B            ESC     EQU     $1B

E000            EPRM.BAS EQU    $E000
2000            VID.BAS  EQU    $2000

00A0            SETWID  EQU     160
0020            SETLEN  EQU     32



                ; RAM usage for data structures

1FFF            STAK.TOP EQU    $1FFF

1E00            TBL.END  EQU    STAK.TOP - 1FFH

1A00            CHAR.TBL EQU    TBL.END - 128 * 8       ; Default character font        *

1900            ESC.TBL  EQU    CHAR.TBL - 256          ; Table of ESC routines         *

18C0            CTRL.TBL EQU    ESC.TBL - 64            ; Table of CTRL code routines   *

1880            CHAR.BUF EQU    CTRL.TBL - 64           ; Buffer for character input

1870            PALT.TBL EQU    CHAR.BUF - 16           ; Copy of palette RAM

1858            INT.TBL  EQU    PALT.TBL - 24           ; Interrupt vector table        *

1840            PKT.Q    EQU    INT.TBL - 24            ; Packet queue
```

```
                ; The following defines zero page RAM usage. RAM is assigned from
                ; the top downwards.

0100    PAGE0     EQU   256
00E0    WORK      EQU   PAGE0 - 32          ; Workspace                    *
00DF    Y.COORD   EQU   WORK - 1            ; Position of text cursor      *
00DE    X.COORD   EQU   Y.COORD - 1         ; Position of text cursor      *
00DE    COORDS    EQU   X.COORD             ; Word pointer to load X and Y *
0000                                        ; together
00DD    CH.COUNT  EQU   X.COORD - 1         ; Number of chars in buffer
00DC    CH.START  EQU   CH.COUNT - 1        ; First character in buffer
00DB    GRAPH.ON  EQU   CH.START - 1        ; Graphic characters
00DA    HIN.ON    EQU   GRAPH.ON - 1        ; Half intensity
00D9    CTRL.ON   EQU   HIN.ON - 1          ; Print control chars
00D8    INVIS.ON  EQU   CTRL.ON - 1         ; Invisible mode
00D7    FLASH.ON  EQU   INVIS.ON - 1        ; Flashing mode
00D6    RVRSE.ON  EQU   FLASH.ON - 1        ; Reverse video mode           *
00D5    UDLIN.ON  EQU   RVRSE.ON - 1        ; Underline mode on
00D4    CUR.OFF   EQU   UDLIN.ON - 1        ; Cursor off                   *
00D3    CUR.CNT   EQU   CUR.OFF - 1         ; Number of VSYNCs to cursor
0000                                        ; flash,  = 0 if not flashing
00D2    CUR.ON    EQU   CUR.CNT - 1         ; =$FF cursor displayed
00D0    CUR.POS   EQU   CUR.ON - 2          ; Absolute address giving      *
0000                                        ; location of first byte of RAM
0000                                        ; for text cursor
00CF    MODE      EQU   CUR.POS - 1         ; Byte indicating current mode: *
0000                                        ; 1 represents 1
0000                                        ; 2 represents 2
0000                                        ; 4 represents 4
00CE    PORT1     EQU   MODE - 1            ; Copy of bits of port 1       *
00CD    PORT2     EQU   PORT1 - 1           ; Copy of bits of port 2       *
00CC    PLOT.MOD  EQU   PORT2 - 1           ; Plotting mode for lines and  *
0000                                        ; points
00CA    Y.PLOT    EQU   PLOT.MOD - 2        ; Y coord for line plotting    *
00C8    X.PLOT    EQU   Y.PLOT - 2          ; X coord for line plotting    *
00C7    PLOT.COL  EQU   X.PLOT - 1          ; Plot colour. Aligned         *
0000                                        ; throughout byte
00C6    SCR.WID   EQU   PLOT.COL - 1        ; 2 * Number of chars wide on  *
0000                                        ; mode 2
00C5    SCR.LEN   EQU   SCR.WID - 1         ; Number of text rows          *
00C3    SCR.END   EQU   SCR.LEN - 2         ; First RAM address after      *
0000                                        ; refresh RAM
00C2    CHAR.WID  EQU   SCR.END - 1         ; Number of character columns  *
00C0    DISP.BAS  EQU   CHAR.WID - 2        ; Refresh RAM base address     *
00BE    DASH.MOD  EQU   DISP.BAS - 2        ; Description of dashed lines
00BD    CHAR.MOD  EQU   DASH.MOD - 1        ; =2 for modes 1,2             *
0000                                        ; =4 for mode 4
00BC    CHAR.COL  EQU   CHAR.MOD - 1        ; Colour to print chars in     *
```

```
00BB            CHAR.BAC EQU    CHAR.COL - 1            ; Background colour for chars   *
00B9            SUB.PTR  EQU    CHAR.BAC - 2            ; Pointer to Submit data        *
00B8            SUB.CNT  EQU    SUB.PTR - 1             ; Count of Submit data          *
00B6            CLOCK    EQU    SUB.CNT - 2             ; Count of number of VSYNCs     *
00B5            PKT.CNT  EQU    CLOCK - 1               ; Number of packets on queue
00B4            HOS.MODE EQU    PKT.CNT - 1             ; = $FF
00B3            PALT.FLG EQU    HOS.MODE - 1            ; =$FF when pallette has changed
00B2            STLI.ON  EQU    PALT.FLG - 1            ; =0 when status line is off    *
00B0            TEX.END  EQU    STLI.ON - 2             ; Bottom of text. ( Above status
0000                                                    ; line )                        *
00AE            PLOT.POS EQU    TEX.END - 2             ; Graphics cursor position      *
00AC            PLOT.WID EQU    PLOT.POS - 2            ; Number of logical pixels wide *
00AA            PLOT.LEN EQU    PLOT.WID - 2            ; Number of logical pixels high *


E000                    ORG EPRM.BAS

                ; EPROM data
E000 01                 DB      VERS                    ; Version number
E001 03                 DB      ISSUE                   ; Issue number
E002 10                 DB      TYPE                    ; Function information

                ; Patch bit 4 of the type byte ( set to 0 ) to disable
                ; the test screen after reset.

                ; Table of routines in EPROM
                ; 1st element stored at EPRM.BAS + 3
E003 7EE0D2             JMP     START           ; Cold start                    *
E006 7EE0D2             JMP     WARM            ; Warm start
E009 7EE1A4             JMP     MAIN            ; Main routine
E00C 7EE1C0             JMP     ESCAPE          ; Escape handler                *

                ; OR or AND Reg A with PORT2
                ; Use this for all modifications to PORT2
                ; Modifies WORK+14,WORK+15
E00F 7EE34B             JMP     ORP2                                            ;*
E012 7EE35E             JMP     ANDP2                                           ;*

                ; Send byte to host
                ; Sends a byte to the host. Will not put the next byte until
                ; the last one has been read
E015 7EE332             JMP     PUTIT                                           ;*

E018 7EE322             JMP     ERROR           ; Send error byte to host

                ; Get a byte from the input stream.
                ; Value returned in A. Regs B and X preserved.
                ; See also READ and READHX
E01B 7EE298             JMP     GETCH           ; Get char from input stream    *
E01E 7EE39F             JMP     WRCH            ; Write char to screen          *
E021 7EE4F7             JMP     PRCH            ; Print char without updating cursor *
```

```
                ; Move a block of data from address in WORK+2 to address in WORK
                ; Number of bytes to copy in Reg D.
                ; UMOVE copies from given addresses upwards through memory and DMOVE
                ; downwards.
E024 7EE271             JMP     UMOVE                                   ;*
E027 7EE24A             JMP     DMOVE                                   ;*

                ; Calculates useful addresses
                ; Calculates CUR.POS from DISP.BAS, X.COORD and Y.COORD, returned in WORK
                ; Address of first character in row returned in WORK+2
                ; WORK+4 to WORK+15 may be used for other values
E02A  .3CB              JMP     SCREEN          ; Calculate CUR.POS, return in WORK     *

                ; Deletes any text cursor at CUR.POS, recalculates CUR.POS from X.COORD,
                ; Y.COORD and handles scrolling, cursor going off the screen etc.
                ; Redraws new cursor if enabled
E02D 7EE417             JMP     CURSOR                                  ;*

E030 7EE55E             JMP     SCROLL          ; Scroll text up one line               ;*
E033 7EE3E4             JMP     SCR.LIM         ; Ensure text cursor is on screen       *
E036 7EE90C             JMP     ERA.SCR         ; Clear screen to pallette colour 0     *

                ; READ reads (Reg B) bytes of data from the input stream and reads them
                ; into WORK+16 upwards. B (= 16.
                ; READHX reads two bytes from the input stream and treats them as two
                ; ASCII hex digits to be converted into one byte value. Value returned
                ; in A. Undefined if two bytes are not hex digits.
E039 7EE829             JMP     READ                                    ;*
E03C 7EE897             JMP     READHX

                ; Convert ASCII hex digit to a nibble
E03F 7EF09C             JMP     HEXBIN                                  ;*

E042 7EF09B             JMP     GSX             ; Handle GSX functions
E045 7EE90B             JMP     LINE.INS        ; Insert line                           *
E048 7EE955             JMP     CHAR.INS        ; Insert char                           *
E04B 7EE929             JMP     LINE.DEL        ; Delete line                           *
E04E 7EE982             JMP     CHAR.DEL        ; Delete char                           *
E051 7EE8CE             JMP     CUR.UP          ; Cursor up                             *
E054 7EE8E0             JMP     CUR.DOWN        ; Cursor down                           *
E057 7EE887             JMP     CUR.LEFT        ; Cursor left                           *
E05A 7EE8D4             JMP     CUR.RITE        ; Cursor right                          *
E05D 7EE88D             JMP     TAB             ; Tabulate                              *
E060 7EE8C8             JMP     LINEFEED        ; linefeed                              *
E063 7EE8DA             JMP     RETURN          ; Carriage return                       *
E066 7EE8EB             JMP     HOME            ; Cursor home                           *
E069 7EE8F4             JMP     NEWLINE         ; Carriage return, linefeed             *

                ; Unsigned multiplies Reg A by value in (WORK+2,WORK+3)
                ; Returned in WORK,WORK+1
```

```
E06C 7EE836          JMP     MULT16                                      ;*

E06F 7EE848          JMP     DIV16          ; 16-bit division
E072 7EE752          JMP     SET.PALT       ; Update pallette
E075 7EF0AF          JMP     SET.REG        ; Set 6845 reg. Reg no in A, Value in B*

E078 7EEAC8          JMP     PLOT           ; Plot point at ((WORK+12),(WORK+14))  *
E07B 7EEB31          JMP     PNT.MASK       ; Plot point at ((WORK+20),(WORK+22)), *
E07E                                        ; clipped and dashed

                ; Plots a line from graphics cursor to point whose X coord is passed
                ; in WORK+12, Y coord in WORK+14 (double bytes)
E07E ..  3C          JMP     LINE                                        ;*

E081 7EEF7F          JMP     ARC
E084 7EEAED          JMP     CHK.LIM        ; Check ((WORK+20),(WORK+22)) is on    *
E087                                        ; screen
E087 7EEB0C          JMP     PNT.LOC        ; Memory location of pixel             *
E08A                                        ; ((WORK+20),(WORK+22)) returned in
E08A                                        ; WORK+0
E08A 7EEB31          JMP     PNT.MASK       ;

E08D 7EE388          JMP     ANDP1          ; AND PORT1 with Reg A. ( Uses WORK + 14*
E090 7EE371          JMP     ORP1           ; OR PORT1 with Reg A.  ( and WORK + 15 *
E093 7EEA1D          JMP     DEL.RITE       ; Erase to end of line                 *
E096 7EEA42          JMP     PAGE.DEL

E099 7EE1D1          JMP     DMOVE8         ; As DMOVE but move COUNT*8 bytes      *
E09C 7EE210          JMP     UMOVE8         ; As UMOVE but move COUNT*8 bytes      *

E09F 7E  D5          JMP     LEFSCR                 ; Scroll whole screen left     *
E0A2 ..  5A          JMP     RIGSCR                 ; Scroll whole screen right    *
                ; Scroll part of one line. Offset from base of display in WORK+16,
                ; Number of bytes to scroll in WORK + 18. ( Both values 2 bytes )
E0A5 7EE6ED          JMP     BLLEFT                                          ;*
E0A8 7EE71B          JMP     BLRIGT                                          ;*

                ; Routines used in ESC 90 command to step graphics cursor one place
                ; and draw point in mode 4 in XOR plotting.
E0AB 7EEFD1          JMP     PLOT.INC
E0AE 7EF085          JMP     INC.PLOT               ; XOR plot at cursor           *
E0B1 7EEFE1          JMP     INC.1                  ; Move up and right and plot   *
E0B4 7EF02F          JMP     X.INC                  ; Move right                   *
E0B7 7EEFF0          JMP     INC.3                  ; Move down and right          *
E0BA 7EF055          JMP     Y.DEC                  ; Move down                    *
E0BD 7EEFFF          JMP     INC.5                  ; Move down and left           *
E0C0 7EF042          JMP     X.DEC                  ; Move left                    *
E0C3 7EF00E          JMP     INC.7                  ; Move up and left             *
E0C6 7EF06D          JMP     Y.INC                  ; Move left                    *

                ; Fill area of screen in plotting colour.
```

```
                ; Two corners of rectangle given by (X.PLOT,Y.PLOT) and (WORK+12,WORK+14)
E0C9 7EEE2E             JMP     FILL                                            ;*

                ; Draw line in XOR mode 4 pixels. Line must be on screen.
                ; Draws from graphics cursor to (WORK+12,WORK+14)
E0CC 7EED58             JMP     QLINE                                           ;*

                ; Read pallette colour of pixel at coords (WORK + 12, WORK + 14)
                ; Value sent back to host
E0CF 7EEA03             JMP     RD.PNT                                          ;*
```

SGT1 COLOUR CODING

	D7	D6	D5	D4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
D3					0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
D2					0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
D1					0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
D0					0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	0	0	0		BL	R	G	Y	BU	M	C	W	BL	R	G	Y	BU	M	C	W
0	0	0	1		R/BL	R	Y/G	Y	M/BU	M	W/C	W	R/BL	R	Y/G	Y	M/BU	M	W/C	W
0	0	1	0		G/BL	Y/R	G	Y	C/BU	C/M	C	W	G/BL	Y/R	G	Y	C/BU	C/M	C	W
0	0	1	1		Y/BL	Y/R	Y/G	Y	W/BU	W/M	W/C	W	Y/BL	Y/R	Y/G	Y	W/BU	W/M	W/C	W
0	1	0	0		BU/BL	M/R	C/G	W/Y	BU	M	C	W	BU/BL	M/R	C/G	W/Y	BU	M	C	W
0	1	0	1		M/BL	M/R	W/G	W/Y	M/BU	M	W/C	W	M/BL	M/R	W/G	W/Y	M/BU	M	W/C	W
0	1	1	0		C/BL	W/R	C/G	W/Y	C/BU	C/M	C	W	C/BL	W/R	C/G	W/Y	C/BU	C/M	C	W
0	1	1	1		W/BL	W/R	W/G	W/Y	W/BU	W/M	W/C	W	W/BL	W/R	W/G	W/Y	W/BU	W/M	W/C	W
1	0	0	0		BL	R	G	Y	BU	M	C	W	BL	R	G	Y	BU	M	C	W
1	0	0	1		R/BL	R	Y/G	Y	M/BU	M	W/C	W	R/BL	R	Y/G	Y	M/BU	M	W/C	W
1	0	1	0		G/BL	Y/R	G	Y	C/BU	C/M	C	W	G/BL	Y/R	G	Y	C/BU	C/M	C	W
1	0	1	1		Y/BL	Y/R	Y/G	Y	W/BU	W/M	W/C	W	Y/BL	Y/R	Y/G	Y	W/BU	W/M	W/C	W
1	1	0	0		BU/BL	M/R	C/G	W/Y	BU	M	C	W	BU/BL	M/R	C/G	W/Y	BU	M	C	W
1	1	0	1		M/BL	M/R	W/G	W/Y	M/BU	M	W/C	W	M/BL	M/R	W/G	W/Y	M/BU	M	W/C	W
1	1	1	0		C/BL	W/R	C/G	W/Y	C/BU	C/M	C	W	C/BL	W/R	C/G	W/Y	C/BU	C/M	C	W
1	1	1	1		W/BL	W/R	W/G	W/Y	W/BU	W/M	W/C	W	W/BL	W/R	W/G	W/Y	W/BU	W/M	W/C	W

BL = BLACK
R = RED
G = GREEN
Y = YELLOW
BU = BLUE
M = MAGENTA
C = CYAN
W = WHITE